ActiveState®

PYTHON MODERNIZATION: 2 TO 3

5 STEPS TO SUCCESS

1. **1.** Upgrade your existing code base to at least Python 2.6 (Python 2.7 preferred).
2. **2.** Decide whether you need to continue supporting Python 2, or if you can just go straight to Python 3.
3. **3.** Ensure you have a robust test suite in place to continually test for compatibility.
4. **4.** Ensure you have a good understanding of Python 3 conventions.
5. **5.** Choose your tooling:

   Six: best for adding Python 3 compatibility to your existing Python 2 code.

   2to3: best for converting Python 2 code to Python 3 code.

   Python-future: best for those that want to focus on writing python 3 code going forward while ensuring compatibility with Python 2.

Migrating from Python 2 to Python 3

Python 2 has two key problems because it was created before the Unicode standard was finalized. One of the consequences of its early implementation is that Python 2 does not strictly enforce Unicode conventions. As a result, it's inappropriate for coding in non-Roman languages. The other consequence is that Python 2 implements a string type that does not clearly separate textual data from binary data. This lack of clarity often leads to erroneous bugs being introduced into the code.

Python 3 resolves the two key issues with Python 2 by basing the default core string type on Unicode. This is a big gain for Python programmers because it eliminates the text/binary confusion and enables multilingual programming. Python 3 also introduces a number of other backwards-incompatible changes, including:

- Removal of classic Python 2 classes
- Changing integer division to automatically generate a floating point result
- Converting the print statement to a function

THE FUTURE OF PYTHON

Python 2.7 has been officially declared as the last version of Python 2 and it will cease to be actively maintained on January 1, 2020. Further, Python 2.6 is no longer supported. This has significant ramifications for enterprises with applications running under Python 2 as they'll need to migrate to Python 3 sooner rather than later.

Python is one of the most popular programming languages in the world, driven primarily by ease of use and the number of tasks, like machine learning, for which it is ideally suited. As the number of new Python users grows, they're most likely to start coding with their default Python installation, which is more and more likely to be Python 3.

PYTHON MODERNIZATION: 2 TO 3

THE FOLLOWING STEPS PROVIDE AN OVERVIEW OF THE MIGRATION PROCESS:

1. Upgrade your code base to at least Python 2.6 (Python 2.7 preferred).
2. Decide whether you need to continue supporting Python 2, or if you can just go straight to Python 3.
3. Ensure you have a robust test suite in place to continually test for compatibility.
4. Ensure you have a good understanding of Python 3 conventions.
5. It's time to pick your tooling. There are currently three recommended packages:

SIX
Best if you want to add Python 3 compatibility to your existing Python 2 code. Six provides a set of utilities that wrap over the differences and allow you to run your code under both Python 2 and Python 3.
Pros: as a single Python file, six can be easily copied into your project.
Cons: need to remember to import packages from the six.moves library since Python 3 moved several functions to different modules.

2TO3
Best for converting Python 2 code to Python 3 code. It reads Python 2 source code and applies a series of fixers to transform it into valid Python 3 code.
Pros: can be rigged to run automatically; can be extended to handle corner cases.
Cons: imperfect conversion; assumes you want to continue coding in Python 2.

PYTHON-FUTURE
Much like six, it provides a compatibility layer for Python 2 and 3 code, but python-future is intended for those that want to focus on writing python 3 code going forward.
Pros: unlike six, python-future allows you to issue standard import commands.
Cons: Assumes you only want to write Python 3 code.

BENEFITS TO MIGRATING BEFORE THE DEADLINE

Improved Compatibility
94% of the top 360 most downloaded packages offer Python 3 support.

Ease of Porting
Python 3.5 introduced changes that simplified code porting in Sept 2015.

Cloud Support
The three largest public cloud providers (AWS, Azure and Google Cloud Platform) fully support Python 3.

OS Support
Many versions of Linux (including Ubuntu and Fedora) now install Python 3 by default.

Talent Pool
Your Python 2 programmers can grow their skills to work on new corporate projects.

Recruitment
Be an attractive workplace to additional Python programmers.

BEST PRACTICES IN PORTING

You have a number of choices for migrating from Python 2 to Python 3. The recommended course of action is to modernize incrementally in order to address failures progressively, rather than being overwhelmed by the task/errors all at once. E.g. over multiple releases of your application

ACTIVESTATE – PYTHON BUILDS SINCE 1999

It's always a good idea to have an expert to help smooth the way. ActiveState is the de-facto standard for millions of developers around the world and companies like IBM, Bombardier and CapitalOne. ActiveState has been providing commercially-backed, secure, stable and comprehensive OSS language distributions for over 20 years including critical SLAs and maintenance updates. ActiveState's open source language distributions can also be freely downloaded and are 100% compatible with community open source code.

ActiveState®
website: www.activestate.com
Toll-free in NA: 1.866.631.4581
email: solutions@activestate.com